PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F A2

(11) International Publication Number: WO 00/57258

(43) International Publication Date: 28 September 2000 (28.09.00)

(21) International Application Number: PCT/US00/06924

(22) International Filing Date: 17 March 2000 (17.03.00)

(30) Priority Data:

60/125,594	19 March 1999 (19.03.99)	US
09/444,530	22 November 1999 (22.11.99)	US

(81) Designated States: AU, BR, CA, JP, SG, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(71) Applicant: CYBERSOURCE CORPORATION [US/US]; Suite 301, 550 S. Winchester Boulevard, San Jose, CA 95128 (US).

(72) Inventors: ARNOLD, Tom; 14697 Carnelian Glen Court, Saratoga, CA 95070 (US). LEWIS, Michael, J.; 241 Arriba Drive, #1, Sunnyvale, CA 94086 (US). MARTIN, Todd; 5665 Begonia Drive, San Jose, CA 95124 (US).

(74) Agents: PALERMO, Christopher, J. et al.; Hickman Palermo Truong & Becker, LLP, 1600 Willow Street, San Jose, CA 95125–5106 (US).

(54) Title: METHOD AND APPARATUS FOR VERIFYING ADDRESS INFORMATION

(57) Abstract

A method and apparatus are provided for automatically verifying address information. A consumer places an online order for a product by providing order information to an electronic commerce system through a merchant and over a public data network. The order information includes a delivery address to which the consumer desires the product to be delivered. The delivery address is parsed and converted, using a plurality of tests, to create a hypothesized address that is formatted according to a standard format. The tests includes City–State–Zip confirmation and Street Address parsing. The hypothesized address is looked up in an address database. If the lookup fails, other tests are carried out based on assumptions about possible errors in the results of the previous tests and in the address information. One or more further lookups are carried out to verify the address. Information that describes the results of these operations may be passed to a calling program, enabling it to interact with the consumer to correct any errors in the delivery address. As a result, the accuracy of deliveries of products ordered through online electronic commerce systems is significantly improved.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

METHOD AND APPARATUS FOR VERIFYING ADDRESS INFORMATION

The application claims priority from prior U.S. Provisional Patent Application Serial No. 60/125,594, filed on March 19, 1999, entitled METHOD AND APPARATUS FOR VERIFYING ADDRESS INFORMATION, which is hereby incorporated by reference in its entirety as if fully set forth herein.

FIELD OF THE INVENTION

The present invention generally relates to data processing. The invention relates more specifically to methods, apparatus, and products that provide automatic intelligent verification of address information.

BACKGROUND OF THE INVENTION

Electronic commerce systems, in which buyers can order and pay for products online, have gained wide use throughout the world. Some electronic commerce systems may be used to process transactions with individual consumers, and others may be used to carry out transactions between businesses, known as business-to-business electronic commerce.

The electronic commerce systems that interact with consumers generally collect online orders submitted by the individual consumer over a public data network. Normally, a consumer who wishes to order a product fills out and submits an online form, or answers a series of prompts from the electronic commerce system. The electronic commerce system sends the order information to a merchant that fulfills the order. Unfortunately, consumers do not provide accurate order information.

A particular problem arises when a consumer submits an order that includes a product delivery address or "ship to" address that does not actually exist. This problem may arise through data entry error, omission of required information, lack of knowledge on the part of the consumer, or deceit. An electronic commerce system may collect a valid order with an invalid address, so that the product cannot be delivered successfully to the delivery address. As a result, the product is returned to the sender even when the order is valid.

The merchant who fulfills such an order incurs both direct and indirect costs from this problem. Direct costs include loss of shipping and handling fees, paying shipping charges on the return of the product, charges for restocking, charges for investigation of the problem, and charges associated with re-shipping the product to the correct address. Indirect costs are harder to quantify; however, they may have greater qualitative cost because they may adversely affect customer satisfaction with the merchant. The consumer believes that all the information was entered correctly, and that the merchant accepted the order. The merchant believes that the order is valid, but is unable to satisfy the consumer by shipping the product to the correct address.

Thus, there is a need for a way to automatically examine delivery information and verify that it represents a correct real-world physical location or postal address, so that a merchant avoids shipping products to invalid addresses.

In the past, others have attempted to address this problem by setting up off-line systems that receive a batch of order data and seek to verify the address information by comparing it against "known good" information. However, such off-line batch-mode systems are impractical for use in the fast-paced online electronic commerce environment. In particular, there is a need to provide immediate, real-time online feedback to the consumer when an address error is detected, so that the consumer can correct and re-submit the address before an order is processed.

It is especially desirable to provide the feedback online, while the consumer is entering information in an order form or similar document.

Thus, in the context of online electronic commerce, there is a particular need to automatically examine delivery information submitted online by an individual or business consumer, and verify in real-time, without unacceptable processing delay, that the delivery information contains a valid real-world physical location or postal address.

Past attempts to solve this problem are also deficient in failing to adequately interpret addresses that are expressed in a language other than English. Significant electronic commerce business is carried out by companies in North America that ship to addresses in Canada. Addresses in the Province of Quebec, as well as addresses in certain regions of the United States such as Louisiana, may be expressed in French. Thus, there is also a need for a way to verify addresses that are expressed using French words and grammar, as well as addresses in other languages.

Prior approaches to processing of address information include online map-generation and direction-finding services, such as Mapquest and Zip2 (www.zip2.com). These services parse address information in order to generate driving directions or generate a map. However, these services do not attempt to guess the meaning of ambiguous address information and generally carry out simple, unsophisticated parsing. The reason is that these services are free, and merely generate information, and therefore the risks and costs of failure or error are negligible. If the service cannot understand an address, it simply informs the user that the address is not understood or contains an error, and the user is expected to re-enter the address. This dialogue takes place using HTTP messaging over the World Wide Web, thereby providing an interface to the user.

In contrast, in the online commerce environment, the cost of erroneous delivery is high, and therefore there is a need for highly accurate, sophisticated parsing of address information. There is no user interface and no opportunity to obtain further information from the user. It is undesirable to tell the end user that the address is wrong without making every possible attempt to verify the address, because the end user may revoke the online commerce transaction or become frustrated with the sponsoring merchant and go elsewhere.

SUMMARY OF THE INVENTION

The foregoing needs and objects, and other needs and objects that will become apparent from the following description, are achieved by the present invention, which comprises, in one aspect, a method for automatically verifying address information. A consumer places an online order for a product by providing order information to an electronic commerce system through a merchant over a public data network. The order information includes a delivery address to which the consumer desires the product to be delivered. The delivery address is parsed and converted, using a plurality of tests, to create a modified address that is formatted according to a standard format. The tests include City-State-Zip confirmation and Street Address parsing. The modified address is looked up in an address database. If it is found, a response message indicating success may be passed to a calling program so that the order is entered and fulfilled.

If the lookup fails, other tests may be carried out based on assumptions about possible errors in the results of the previous tests and in the address information. One or

more further lookups may be carried out to verify the address. Information that describes the results of these operations may be passed to a calling program, enabling it to interact with the consumer to correct any errors in the delivery address. As a result, the accuracy of deliveries of products ordered through online electronic commerce systems is significantly improved.

The invention also encompasses an apparatus, system, and computer-readable medium that may be configured to carry out the foregoing steps.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

FIG. 1 is a block diagram of an electronic commerce system;

FIG. 2 is a block diagram of an address verification mechanism;

FIG. 3 is a block diagram of general processing steps of a City-State-Zip verification mechanism;

FIG. 4 is a block diagram of a computer system with which an embodiment may be implemented.

FIG. 5 is a block diagram of a Street Address verification mechanism;

FIG. 6 is a block diagram of an Address Lookup verification mechanism;

FIG. 7 is a block diagram of general processing steps of a Street Address verification mechanism; and

FIG. 8 is a block diagram of a City-State-Zip verification mechanism.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and apparatus for automatically verifying address information is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

OPERATIONAL CONTEXT

FIG. 1 is a block diagram of an electronic commerce system that represents a context in which an embodiment may be practiced.

Client 102, which executes browser 104, is coupled logically to network 106. Client 102 is any network end station, such as a personal computer, workstation, or other device having a processor. Browser 104 is one or more software or hardware elements that cooperate to read and display electronic documents that are formatted according to open protocols. An example of a commercial product that may be used to implement browser 104 is Netscape Navigator®. Network 106 is a collection of one or more devices and interconnecting elements that support data communications using open protocols. In one embodiment, network 106 is a public, packet switched data network such as the Internet.

A merchant server 108 that executes a merchant application 110 is also logically coupled to network 106. Merchant server 108 is one or more hardware or software elements that provides a point of contact between a user of client 102 and a merchant that is in the business of selling one or more products or services. Merchant server 108 may be located at the merchant's place of business, but this is not required. Normally merchant server 108 is a secure commerce server suitable for use with the World Wide Web, and also includes an HTTP (Web) server. Microsoft® Internet Information Server is an example of a commercial product that is suitable for use as merchant server 108.

Merchant application 110 is one or more hardware or software elements that cooperate to offer products or services to the consumer, display information about the products or services, and solicit orders for the products or services. The merchant application 110 generally provides the main interface of the merchant to the consumer or user. The merchant application 110 may retrieve and store data about products, services, consumers, and orders in a database 111 that is logically coupled to the merchant server 108 and the merchant application.

Commerce server 112 is one or more hardware or software elements that service requests of clients to carry out commercial processes or transaction processes. It may be remote from merchant server 108 or co-located with the merchant server. In the preferred embodiment, commerce server 112 is remote from merchant server 108 and

communicates with the merchant server through network 106 using an agreed-upon protocol.

Commerce application 114 is one or more software elements that cooperate to provide commercial services and transaction services to merchant server 108 and merchant application 110. The commerce application 114 may store and retrieve information using a database 115. An example of an embodiment of commerce application 114 is Internet Commerce System (ICS), a set of on-demand commerce applications that are commercially available from CyberSource Corporation, San Jose, California. The applications carry out services for each transaction point involved in accepting and fulfilling an order.

Commerce application 114 may include gatekeeping applications and fulfillment applications. Examples of gatekeeping applications include a fraud screen that allows a merchant to determine the level of risk it wishes to accept in an order; an export and distribution control application; payment processing applications for receiving and authorizing credit card payments; tax calculation applications; and others. Examples of fulfillment applications include a secure fulfillment messaging application and digital content delivery applications.

Address verification mechanism 116 is an instance of commerce application 114 that carries out address verification and can interact with other commerce applications. Generally, address verification mechanism 116 assesses the existence of addresses in Canada and the U.S. and can verify the end user's deliverable address for physical shipment. Address verification mechanism 116 may also be coupled to and interact with its own database 118 of address information.

Database 118 is a commercial database of standard address information. In one embodiment, database 118 is the USPS Zip+4 National File database that is commercially available from the U.S. Postal Service. The database may contain address information for any location or country. There may be multiple databases, for example, one database for each country of interest.

OVERVIEW OF ADDRESS VERIFICATION METHOD

FIG. 2 is a block diagram of processing steps that may be carried out by an embodiment of address verification mechanism 116.

Generally, a client supplies address information to a server, as shown by block 202. In the preferred embodiment, block 202 involves a consumer providing address information to a merchant or a merchant server. In this context, "consumer" refers to an individual as well as to a business entity such as a corporation, partnership, or other legal person. Normally, the address information represents a location to which a product ordered by the consumer is to be delivered, and therefore the address information may be referred to herein as "delivery address information" or simply "delivery address." However, this is not required. The address information could be provided for use in an application that generates driving directions or maps. The address information could be provided for use in opening an account of some kind, as with an online bank. The address information could be provided in any other context in which address information is processed by automatic data processing equipment.

Normally, the address information includes one or more street address values, a city value, a state or province value, a ZIP code value or other postal code value, and a country value. An example of fields that may comprise the address information is set forth in Table 1 below. In one embodiment, the state value, ZIP code value, and country value are expected to be received by the address verification mechanism 116 in canonical, standard format. The merchant application 110 is expected to scan these values after they are entered. If errors are found, the merchant application 110 corrects the values by enforcing standards for the values. For example, the state value must conform to the two-character state codes used in U. S. Postal Service standards, and the country value must be an ISO-compliant country code. Alternatively, the merchant application 110 prompts the user to re-enter the values.

In one embodiment, the address information may also include a set of shipping address information and separate billing address information. If the shipping address information is missing a city value or state value or country value, but values for the missing fields are provided in the billing address information, then the corresponding fields in the shipping address information are assumed to have the same values. In this embodiment, if the shipping address information includes a city value or state value or country value but is missing a street address value, an error signal is generated. Thus, a particular element of shipping address information may be ignored if less specific information about the shipping address is missing.

As shown by block 204, the server sends the address information to an address verification mechanism for verification of the address information. In the preferred embodiment, the merchant passes the address information to an address verification mechanism for processing.

When the address information is completely processed, results of the processing are returned, as indicated by block 206. The results may include a message that confirms the validity of the address information, or a message indicating that the address is invalid. The results may also include suggestions about possible ways to fix the address, if it is invalid. The server carries out responsive processing, as shown by block 208. If the address verification mechanism 116 has generated an error signal, then the responsive processing may involve displaying an error message to the consumer. Alternatively, the responsive processing may involve carrying out further steps in an online commerce application or other application.

The address verification mechanism generally comprises a City-State-Zip verification mechanism 210, a Street Address verification mechanism 212, and a Lookup mechanism 214. Each such mechanism may be implemented in the form of one or more computer programs, processes, subroutines, methods, threads, or other software elements. Alternatively, the mechanisms may be implemented in data processing equipment or other electronic hardware. Lookup mechanism 214 is logically coupled to database 118, which may be a Sybase® database server, Oracle® database server, or equivalent database system.

In an alternative embodiment, address verification mechanism 116 may be called from merchant server 108 using information that indicates that the merchant does not wish to ship a product to a high-risk address. In one embodiment, a flag value is used to indicate that the merchant does not wish to ship to high risk addresses. In this context, a "high risk address" is an address which is difficult to positively associate with a real consumer. Examples of high risk addresses include mail forwarding addresses, and addresses of private businesses that offer mailbox rentals to consumers. In one embodiment, a pre-defined table of high risk addresses is stored, and the table may be in a database. When the flag value is set, after the city-state-verification mechanism executes, address verification mechanism 116 searches the high risk address table and

seeks to match the current address information to information in the table. If a match occurs, an error signal is generated.

In one preferred embodiment, address verification mechanism 116 may be implemented in the form of one or more computer programs, subroutines, or other software elements that cooperate to carry out the foregoing functions. In this embodiment, address verification mechanism 116 carries out the following general processing steps, as illustrated in FIG. 7.

At block 700 of FIG. 7, the mechanism tests the received address information to ensure that it contains a value for all required values. In one embodiment, the required values are, for a billing address: address1; city; state; ZIP code; country. Optionally, the user may supply a second address line (address2) and a separate shipping address. If the shipping address is provided, it is processed in place of the billing address.

Next, at block 702, the mechanism stores an entry identifying the processing transaction in a log file, and generate an error signal if there is a problem logging the transaction.

At block 704, the mechanism determines whether the address information includes a Shipping Address value and a Billing Address value. If so, then the Shipping Address value is used as indicated by block 708. Otherwise, at block 706, the mechanism uses the billing address.

At block 710, the mechanism cleans the received address information by removing illegal punctuation marks and other symbols.

At block 712, the mechanism tests whether the city and state information in the address information is correct, by looking up the ZIP code value of the address information in the database, retrieving all associated city and state names from the database, and matching the address information to the associated city and state names. If there is no match, then at block 714, the mechanism skips further processing. Optionally, the mechanism takes the true city name value that was obtained from the database in the preceding step and stores it in the city name field of the address information. If there is a match at block 712, then control passes to block 716 for further processing.

At block 716, the mechanism parses the first address line value of the address information to separate it into a set of individual address values, such as street name, street number, pre-directional element, post-directional element, apartment number, etc.

At block 718, the mechanism gets all address records from the database that match the ZIP code of the address information, and at block 720, the checks the address records retrieved from the database against the parsed address line. Alternatively, at block 718, the mechanism first retrieves address records for just those street names that match the ZIP code, and if no match is successful at block 720, retrieves all address records that match the ZIP code and attempt a match.

In one embodiment, the mechanism generates a response message that indicates either that the mechanism successfully found the address information in the database after parsing, as shown at block 722, or that the address information did not match, or that some other error occurred, as shown at block 724. In an alternate embodiment, the mechanism returns a copy of the address information that is corrected and that matches the information in the database, at block 724.

In still another alternate embodiment, the mechanism may generate geo-codes. Based on the parsed address information, the mechanism may generate a longitude value and a latitude value that correspond to the address information. This may be carried out by storing the longitude and latitude values in database 118 and retrieving them when the parsed address information is looked up in the database. This information enables a merchant to correlate goods that are purchased with a specific geographic location or region, and therefore it is usable in carrying out market demographic studies. Alternatively, the mechanism may generate map location values that are keyed to third party printed maps or map books.

The following sections describe some of the foregoing steps in further detail.

CITY-STATE-ZIP VERIFICATION MECHANISM

FIG. 3 is a block diagram of processing steps that may be carried out by one embodiment of City-State-Zip verification mechanism 210.

In block 301, a ZIP code value is received. Block 301 may involve extracting a ZIP code value from a request message that is sent from merchant application 110 to commerce server 112. The ZIP code that is received is assumed to be correct. It has been

found through experience that checking the ZIP code for errors may introduce an unacceptable delay in processing.

As shown by block 302, the City-State-Zip verification mechanism 210 submits a query to the database 118 to obtain all city names and state names that are associated with that particular ZIP code. In block 304, the City and State values that have been received as input from the consumer or from merchant application 110 are tested against the city and state names obtained from the database 118. Block 304 may involve numerous tests, as described further below.

In block 306, the mechanism tests whether the state values match. If there is no match, then an error signal that indicates an error in the state value is generated, as indicated by block 314.

In block 308, the mechanism tests whether the city values match within a reasonable tolerance, taking into account a variety of possible typographical errors, abbreviations, and the like. If there is a match, then the City and State values are considered to be valid, and processing proceeds to the Street Address verification mechanism 212. A success value may be returned, as indicated by block 310. If there is no match, then an error signal is generated, as shown by block 312.

In the preferred embodiment, City-State-Zip verification mechanism 210 is implemented in the form of a software subroutine. The subroutine returns an output value of "0" if there is no match, and returns a value of "1" and the correct city name if there is a match. The subroutine may carry out the following processing steps, as illustrated in FIG. 8.

At block 800 of FIG. 8, the mechanism connects to the database. This may involve checking where the database handle is opened; saving existing environment variables; getting the database user name and password; logging into the database server; and setting the database.

The mechanism then removes all non-alpha characters from city name at block 802. This prevents the mechanism from generating an error signal, for example, when the input city name is "Mc Lean" and the matching city name is "McLean."

At block 804, the mechanism looks up the ZIP code in the database and receive a set of matching records. At block 806, the input city name value is compared to the city name value of each of the matching records in the database using a string comparison

function. The string comparison function returns the value "0" if the two strings don't match, the value "1" if a match is found, and a value of "2" or greater if it detects only a typographical error in one of two otherwise similar strings. If a match is found, then at block 808, the mechanism returns a success value with the true city name.

Otherwise, if no match is found between the city name and one of the records returned from the database, then the customer or user may have abbreviated the city name to a set of familiar initials, for example, "NY" for New York, "LA" for Los Angeles, etc. Accordingly, at block 810, the mechanism tests whether the city name matches one of a set of known abbreviations in a pre-defined table of city name abbreviations. If there is a match, the mechanism retrieves the corresponding full city name from the table, and substitutes it for the abbreviated city name. The database information is then re-checked for a match at block 812. In addition, at block 810, the mechanism may test the returned city names for spaces. If any returned city name contains spaces, then a test string is constructed, comprising the initial letter of each word in the returned city name. For example, the test string "SLC" is constructed from the city name "Salt Lake City." The test string is compared to the city name information received from the consumer.

If a match is found, control passes to block 808 where the mechanism returns a success value with the true city name. Otherwise, at block 814, the mechanism returns an error value.

Other steps may also be carried out.

STREET ADDRESS VERIFICATION MECHANISM

A. OVERVIEW

Street Address verification mechanism 212 may comprise one or more software elements that carry out a series of tests to determine whether the address information represents a valid street address.

FIG. 5 is a block diagram of processing steps that may be carried out by one embodiment of the Street Address verification mechanism 212.

In block 500, the Street Address verification mechanism 212 determines whether the address information is of a "General Delivery" type. If the address information is of

a General Delivery type, then the processing is complete and the Street Address verification mechanism 212 terminates.

At block 502, the Street Address verification mechanism 212 breaks the address information into components called "atoms". At block 504, the Street Address verification mechanism applies a series of tests and processes to the atoms in order to convert the address information into a form that conforms with a commercial database of standard address information such as database 118. The converted address information is hereafter referred to as a "hypothesized address".

The address information may contain zero or more of a variety of errors, and the tests carried out in block 504 may be configured to identify and respond to such errors, including:

1. Typographical errors.
2. Invalid abbreviations, such as "100 Main Strt" rather than "100 Main St."
3. Variant identifiers for the term "apartment."
4. Missing pre-directional terms or post-directional terms. An example of a pre-directional term is the word "North" in the street name "North First Street." An example of a post-directional term is the term "N.W." in the address "600 13th Street, N.W."
5. Missing street type identifiers, such as "Road," "Street," etc.

Accordingly, in the preferred embodiment, the Street Address verification mechanism makes guesses, during its processing, about the correct words intended by the consumer. The guesses are made because the preferred embodiment has no user interface or other mechanism whereby the user can be queried about what was intended, and in fact this is undesirable because the processing is carried out online and in real-time. Flag values are used to keep track of the guesses that are made, so that a guess can be reversed if later processing determines that the guess is incorrect.

In an embodiment, the Street Address verification mechanism also determines what type of address is represented by the address information. Address types may include: Regular, Apartment, Post Office Box, Rural Route, etc. The address type (hereafter referred to as "Record Type") of the address information may determine how the merchant ships a product to the consumer. For example, certain carriers do not ship to

Post Office Boxes. As a specific example, if a consumer provides a Post Office Box address and instructs the merchant to ship by FedEx® or United Parcel Service, which carriers do not ship to Post Office Boxes, the merchant can reject the order or request the consumer to input corrective information.

At block 506, the Street Address verification mechanism marks any changes that have been made to the atoms when applying the series of tests and processes of block 504. The changes are marked with Boolean flags such as "Guess" flags, "Dash-In-Zero" flags, and "Convert Number" flags.

At block 508, the Street Address verification mechanism operation is complete and a hypothesized address is ready to be processed by the Address Lookup mechanism 214.

B. SPECIFIC IMPLEMENTATION

Generally, the Street Address verification mechanism 212 will parse the address line. In this context, "address line" means the "address1" field of the address information. Preferably, the mechanism reads the address line from its beginning to scan for an apartment number if present, address number, and street pre-directional. Then the mechanism reads the address line from its end to scan for other information, such as apartment number, street post-directional, and street type. When these elements are identified, any other elements remaining are the street name. Preferably, the address line is parsed into the following elements: Street Number; Street Name; City; State; Zip; Country; Record Type (street address, high-rise, rural route, general delivery, or 1-5 (which are used for Canada addresses)); Rural Route Type ("RR", "RT", etc.); Rural Route (numeric value); Apartment Number; Pre-Directional; Post-Directional; P. O. Box; Suffix ("Road," "Street," etc.).

Flag values are used to facilitate operation of the verification mechanism. Preferably, the following flag values are used:

Guess flag -- indicates that the Street Address verification mechanism 212 has guessed that a segment of text after a street word ("st.", "ave.", etc.) is an apartment number (i.e. "100 Main St. F").

Dash-In-Zero flag -- indicates that the mechanism has split the first atom of the address line on a hyphen; normally this indicates an apartment, however, some addresses have hyphens in the street number.

Convert Number flag -- indicates whether any numeric expressions in the address were changed from words ("One," "Seventy-Seven," etc.) to digits.

One Half flag -- indicates whether the address had the expression "1/2" in it.

In one specific embodiment, Street Address verification mechanism 212 may comprise one or more software elements that carry out the processing steps described below.

1. Scan the address line and strip out spaces and extraneous punctuation, for example, quotes, question marks, semicolons, brackets, etc. Legal symbols may include comma, period, pound, ampersand, and others.

2. Test whether the address line is the string "General Delivery." General delivery addresses are used in certain rural locations. If a General Delivery is found, then processing is complete and the mechanism terminates operation.

3. Convert the address line into a vector of data. This may be carried out by breaking up the address line into atoms, in which spaces are taken to represent boundaries of atoms. Each atom is stored in a vector of atoms. Atoms in the vector each have a unique reference number. The lowest reference number is "0". Thus, the first atom in the address line is atom zero.

4. Test the address line for slash symbols ("/"). If a slash symbol is found in the address line, then the address line either contains a "one-half" indication, or a second semantic concept that is most likely a second address line ("Address2"). For example, the latter expression may be found in an address line that has the format "241 Arribba / Back of Building." Thus, if there is text in an atom after a slash, and the text is not part of a 'one-half" indication, and the value of Address2 is null, then the mechanism moves the information after the slash to the Address2 field or variable. Address2 may be ignored because the further information it provides is not normally valuable enough to justify processing it and is not usually the type of data found in postal databases.

5. Scan the address line to convert it. In particular, the mechanism removes any atom that does not contain either an alphanumeric character or a pound sign, replaces such atoms with spaces, and then re-scans the address line again to break it up into a new

set of atoms. For example, if the address line contains a comma, the elements on either side of the comma should be treated as at least two different atoms. If there are no spaces in the address (e.g., "10MainStreet"), then the mechanism tests whether there is a transition from numeric characters to alphabetic characters or the converse. At that point, the mechanism breaks the address line into two atoms.

6. Track which words are changed to numbers by storing an empty, initialized list of the positions of the words.

7. Find number words wherever they appear in the address line. Number words include "Five," "Hundred," etc. Also find words with hyphens where both sides are number words, for example, "Two-Fifty." Such conversions may be undone later if they are incorrect, as described further below.

8. Combine numbers where there were two number words in an address record (like "three hundred") or numbers separated by an "AND" (like "three hundred and three").

9. Process rural routes. Rural route addresses may have only two atoms, such as "RR" and "5". As a result, the mechanism may introduce an error by assuming that "RR" is the street number. Accordingly, special processing is used. This may involve looking for keywords in the address line that indicate a rural route, such as "RR," "RFD," "RT," "Route," and "HC". If a rural route keyword is found, set the Street Type value to "Rural Route," store the Rural Route Type value, and store the rural route number in the Street Number value. Call the Zero-Length-Street subroutine, which is described below.

10. Process Canadian addresses having both street addresses and rural routes. Generally, this involves scanning the address atom to determine where the street address information ends and the rural route information begins, and carrying out processing similar to step 9 above.

11. Process street numbers and apartment indicators. If the first atom contains both numbers and non-numbers, split the line on the hyphen if there is one, or where it switches from numbers to letters if there isn't a hyphen. For example, if the first atom comprises the text "123-A," then normally the "123" element is the street number and the "A" element is an apartment number. Accordingly, the mechanism splits these elements and separately stores them in the Street Number and Apartment Number fields.

12. Further processing of street numbers and apartment indicators. If the first atom starts with a pound sign ("#", as in "#10 Downing Street"), discard the pound sign and put the next atom into a temporary storage location ("$temporary_number"). If the next atom is a dash, discard it. If the next atom is a non-digit, store it in the Apartment Number field and store the value of $temporary_number in the Street Number field. Otherwise, store the value of $temporary_number in the Apartment Number field, and store the current atom as the Street Number field. If there is no dash in the atom identified above, then the number information in the address line is stored as the Street Number value, and the alphabetic characters are stored as the Apartment Number value.

This sequence of steps correctly processes addresses having the forms "#204 - A" (in which "204" is the Street Number and "A" is the Apartment Number, because A can't be the street number) and "#204-200" (in which "204" is the Apartment Number and "200" is the Street Number). If there is no dash ("#204A") then the steps assume that "204" is the Street Number and "A" is the Apartment Number.

13. If the mechanism has not yet encountered a Street Number value, it must be the first atom. Accordingly, the value of the first atom is stored in the Street Number field.

14. If the current first atom is one character long and it does not match any of the pre-defined directional keywords ("SE", "N.W.," etc.), then it must be an apartment number. Accordingly, the value of the current first atom is stored in the Apartment Number field. The current first atom may be the street name, in the case of addresses such as "D Street," however, this case is considered later.

15. Extract pre-directional elements. The address line is scanned twice to identify pre-directional elements. This enables the mechanism to correctly identify an address with one pre-directional element, such as "123 South Market Street," and an address with two pre-directional elements, such as "123 S. E. 104th Street." In the latter example, where there is a space between the elements "S." and "E.", without two scanning steps, the mechanism could assume that "S." is an apartment number, such that the address is "123-S East 104th Street." If two pre-directional elements are found, then a Boolean flag variable is set. This enables the mechanism to detect and later correct an error in identifying two pre-directional elements. For example, consider an address having a street field of "S. West Street." The element "West" is the name of the street

and not a pre-directional. The mechanism would convert the address line into "S. W. Street," then to "SW Street, and then put "SW" in the pre-directional variable. This is incorrect; however, an error will be detected later because the mechanism will be unable to match the parsed elements to a record of the database. If such an error occurs, the mechanism can correct the error by testing whether the flag variable is set.

16. Process address lines with "Floor" designations. The mechanism identifies any text indicating "floor", or the equivalent, at the end of an address line. Since the U.S. Postal Service database does not contain "Floor" designations, such text is removed along with any atom that designates the number of the floor.

17. Next the mechanism tests whether the last atom contains one or more pound signs, and splits the last atom into two or more atoms at each pound sign. This is done to correct a lack of proper spacing in the address line, for example, "Main St.#10".

18. Process apartment numbers. A list of pre-defined apartment keywords is stored. The list may include, for example, values such as "apt", "Apt.", "Apart.", etc. If the second-to-last atom contains an apartment keyword that is in the list, then the last atom must be the apartment number and it is stored in the Apartment Number value. If the mechanism previously found a dash in the street number, then the material after the dash must be part of the street number, rather than the apartment number, since this is a more reliable indicator. Similarly, if the last atom starts with an apartment keyword, anything after that must be the apartment number.

19. Process address lines that have no street numbers. This is an unusual case, which appears to apply primarily to colleges or major corporations having addresses of the form "Cummings Rm. 2" or similar. The mechanism reaches this case when there are no atoms left for processing, and the current value of the Street Number field does not contain a digit. This would occur, using the above example for illustration, because the element "Cummings" would be stored as the Street Number, the element "Rm." would be stored as the Street Name, and the element "2" would be stored as an Apartment Number. In response, the mechanism stores the entire string in the Street Name field.

20. Further processing of apartment designations. The mechanism carries out further steps that process addresses of the form "654 State Street Apt No. 9." If an apartment keyword is found at this point, the mechanism removes that atom and re-scans

the address line. If the first atom is an apartment number indicator (for example, "#" or "No."), then that atom is deleted, and the last atom is stored as the Apartment Number value.

21. Scan the address line to identify elements that indicate Post Office boxes or box numbers alone. Check whether previous steps erroneously placed the element "P.O. Box", or the equivalent, in the Street Number field. If so, clear the Street Number field, and restore the element to the beginning of the line, identify and store the box number in the Box field, and delete the "P.O. Box" element. The "P.O. Box" element will be re-added and stored as the Street Name in the Zero-Length-Street subroutine, which is called in this step.

22. Look for post-directional elements. Two passes are carried out, as with pre-directional elements.

23. Check the last atom for a suffix indicator ("Road", "Street", "Drive"). The suffixes must match a pre-defined list of valid suffixes. Save the original form of the address line in a temporary variable (for example, "$expanded_suffix"). The original form is stored for later use. On occasion, an address line may have the form "204 Green Trail" and is intended to refer to the true address "204 Green Trail Road," but since the Road indicator is missing, the mechanism will read "Trail" as the suffix and erroneously look up "204 Green Trl." The temporary variable provides a way to backtrack later, if this occurs.

24. Look for post-directional elements again. This additional step ensures that the mechanism accurately detects a post-directional element that is written before or after the street suffix. For example, the constructs "Main Street South" and "234 Atlantic S. E. Road" are both handled accurately.

25. If there are multiple atoms left in the vector of atoms for the address line, and the second-to-last atom contains a suffix indicator, a pre-directional, or a post-directional, and the last atom is not associated with any other element, then it is likely to be an apartment indicator. Accordingly, the value of the last atom is stored in the Apartment Number variable. Since the last atom might not be an apartment number, this step is classified as a guess, and the Guess flag is set to record that a guess was made. The second-to-last atom may also be a directional, but this is very rare, and therefore the mechanism does not try to guess that possibility.

26. Convert ordinal street names into numerals, since that's how they are stored in the database. If the only remaining atom is an ordinal value, then the mechanism assumes that it is an ordinal street name, and converts it to a numeral value.

27. Test whether the first few atoms represent a French street name, as used in Quebec, Louisiana, and other locations. Grammatically, French street types ("Rue", "Avenue") appear at the beginning of the line. The mechanism may test the first several atoms against a list of pre-defined street type values.

28. Store the values of any atoms that have not yet been processed in the Street Name field. Thus, the Street Name value is anything left over at this point.

29. If the Street Name is "S.R." then it is expanded to "State Route" to match the database. Similarly, if the Street Name is "Hwy" it is expanded to "Highway."

30. If the Street Name appears to start with the text "N." or "S." and the value of Pre-directional is null, then a pre-directional element was missed, and the mechanism will locate it and store it in the Pre-directional field.

31. If the Street Name is only a number, such as "9", "104", etc., then the mechanism adds to the Street Name a numeric suffix that matches the number, such as "RD", "ND", "TH", etc.

32. If the Street Name contains multiple spaces, change them to one space, and delete any leading or trailing spaces.

33. Delete any characters in the Apartment Number field that are not alphanumeric.

34. Determine the Record Type of the address:

a. If Apartment Number field is not null, then the Record Type is set to "H" for "High-rise". In that case, the mechanism determines the Apartment locator code that is used by the U.S. Postal Service for the current address.

b. If the Box field is not null and the Street Name field is "P.O. Box," then the Record Type is set to "P" for "PO Box."

c. If the Rural Route field is not null then the Record Type is set to "R" for "Rural Route."

d. If Record Type is null at this point, set Record Type to "S" for "Street Address."

e. If the value of the Country field is "Canada," then the Record Type value is converted to a slightly different value to match the Canada database.

C. PROCESSING ZERO LENGTH STREET NAMES

In the preferred embodiment, the Street Address verification mechanism 212 includes a sub-process or subroutine for processing addresses in which the above-described process disposes of all atoms in the original address, but the Street Name value is empty or has a length of zero. In this case, several tests may be used to discover the correct address. The subroutine may be called whenever an atom is deleted from the original address information. For example, the subroutine may be called during processing of Rural Route addresses.

Recall that the Guess flag indicates that the Street Address verification mechanism 212 has guessed that a segment of text after a street word ("st.", "ave.", etc.) is an apartment number. If the Guess flag is set, then the guess can be undone, and the street name processed again.

If the Pre-directional field contains a value, then it is possible that its value is erroneous and the street name is a compass direction. The Pre-directional field may be cleared and the Street Name field may be set to the directional word associated with the value of the Pre-directional field.

If the Post Office Box field contains a value, then the street name is set to "PO Box".

In another possibility, the Apartment Number value may be the true street name. This may occur, for example, in processing street addresses that involve one-letter street names ("123 D Street"). Previous processing steps may assume that the "D" element is an Apartment Number rather than a street name.

If the Street Number value contains alpha characters, then it may be the true street name. This could occur where the addressee is the only location on the street. In response, the mechanism may store the Street Number value in the Street Name field, and clear the Street Number field.

If the original street address contained only two elements and one of them is a valid suffix word, for example, "204 Mill" or "105 Falls," then previous processing

may have stored "Mill" in the Suffix field. Accordingly, the Suffix field may be stored in the Street Name field and the Suffix field may be cleared.

If none of the foregoing tests identifies meaningful information in a field other than Street Name, then the subroutine may generate an error signal or return a failure code or error message.

D. DATABASE CHECK

In the preferred embodiment, the Street Address verification mechanism 212 includes a sub-process or subroutine for checking received values against the database. Preferably, separate database inquiries are carried out for United States addresses and Canada addresses. This may be carried out by testing the value of the Country field and selecting either a United States address database or a Canada address database. Each check may involve checking the database for appropriate address records and return them by loading a command buffer with appropriate SQL statements and executing the command buffer.

ADDRESS LOOKUP MECHANISM

A. OVERVIEW

FIG. 6 is a block diagram of processing steps that may be carried out by one embodiment of the Address Lookup mechanism 214. At block 600, the Address Lookup mechanism 214 retrieves address records from database 118 by submitting queries to database 118 to obtain all available information about streets having names that match the street name identified by the Street Address verification mechanism 212, and having a ZIP code that matches the ZIP code provided by the consumer.

At block 602, the Address Lookup mechanism 214 performs a first set of tests that checks each address record that is returned from the database against the hypothesized address produced by the Street Address verification mechanism above. For each address record, the Address Lookup mechanism performs the first set of tests to determine whether the address information is valid. The first set of tests is described in the next subsection entitled, "Specific Implementation", under items 2 through 6.

If all the tests succeed, then the Address Lookup mechanism 214 stores a success value associated with the current address record. The success value may be a Boolean

flag value that is stored in a Boolean vector, in which each element of the Boolean vector corresponds to an address record in the database.

At block 604, the Address Lookup mechanism 214 determines if there is a matching address record. At block 606, the Address Lookup mechanism 214 determines if there are multiple matching address records. If there is only one matching address in the database, then at block 608, the Address Lookup mechanism returns a success signal or a message that indicates that the address information has been confirmed.

Otherwise, if there are multiple matches, then at block 612 a second set of tests are carried out. This second set of tests is described in the next subsection entitled, "Specific Implementation", under item 8. In these secondary tests, the address information is re-processed against each matching address record, to ensure that the address information does not contain any ambiguity that could result in misdelivery of the product to the consumer. At block 616, the Address Lookup mechanism 214 determines if there is a matching address record. If there are no matches, then at block 622, the Address Lookup mechanism returns an error signal. Otherwise control passes to block 608.

Returning to block 604, if the Address Lookup mechanism 214 determines there is a no matching address record then at block 610, the Address Lookup mechanism performs a third set of tests. The third set of tests is described in the next subsection entitled, "Specific Implementation", under items 9 through 13. In the third set of tests, the Address Lookup mechanism may selectively reverse one or more guesses that were made by the Street Address verification mechanism 212, and then re-check the address records that were returned from the database.

At block 614, the Address Lookup mechanism 214 determines if there is a matching address record. If there is a matching a record, control passes to block 606, which is described above. If the third set of tests fails, then the database is accessed a second time. When the third set of tests fails, at block 618, the Address Lookup mechanism performs a fourth set of tests. The fourth set of tests is described in the next subsection entitled, "Specific Implementation", under items 14 through 18.

In the second access to the database, the Address Lookup mechanism requests all address records that correspond to the ZIP code of the address information. Generally, the second access step represents a last alternative which normally indicates that the

street name is misspelled. Thus the fourth set of tests include checking whether the street name specified in the address information is misspelled or otherwise contains a typographical error.

At block 620, the Address Lookup mechanism 214 determines if there is a matching address record. If there is a matching a record, control passes to block 606, which is described above. If there is no matching address record then the Address Lookup mechanism 214 returns an error signal at block 622.

In an alternate embodiment, the Address Lookup mechanism or another element of Address Verification mechanism 116 can generate a corrected address string.

B. SPECIFIC IMPLEMENTATION

In one embodiment, Address Lookup mechanism 214 is implemented as one or more software sub-programs or subroutines that check information in database 118 against the parsed address line as represented by the field values described above. In the preferred embodiment, the Address Lookup mechanism 214 is a subroutine that has the following local variables:

$tempst is used to remember the Street Name value before test manipulations are carried out.

$suffixlist, $sfxlist2, $tempsuffix are used to find out if a given Street Name value has multiple suffixes associated with it. If not, the mechanism may ignore the Suffix value.

$predirlist, $prelist2, $temppredir, $postdirlist, $postlist2, $temppostdir are used to find out if a given Street Name has multiple pre- or post- directionals associated with it. If not, the mechanism may ignore the Pre- or Post-directional value.

$tempstreet stores the Street Name value without any intervening spaces, because this is the preferred database format. This avoids having a street name of "De Anza" that does not match the word "Deanza" in the database.

$sufxadd is used for tracking whether we have added the suffix back onto the street name.

Preferably, in this embodiment, Address Lookup mechanism 214 involves the following processing steps:

1. Retrieve all address records in the database that have the same values as the Street Name field and ZIP Code field that have been created in the foregoing parsing steps.

2. Based on the type of the current record, test the field values against each returned address record from the database. In one embodiment, the tests involve comparing each field value against an associated value without giving effect to spaces. Other tests may involve, for example: check that the address is between the high and low ranges returned by the database for that street, check that the pre-directional value and post-directional values match if they are present, check that the odd/even code is the same, etc. A test vector is established and stored in association with the current record. The test vector has one bit corresponding to each of the tests described above, and all bits are initialized to True. If any test fails, then a Boolean value or bit associated with the test is set to False.

3. Special processing for Post Office Box entries (Record Type "P"). The preferred database includes an entry for the lowest and highest valid P.O. box number for a particular ZIP code. Due to data entry errors and data gathering errors, some database records comprise a lower box number that is numeric (e.g., "200") and a highest box number that is alphabetic (e.g., "F"). In response, the mechanism assumes that the sequence of numbers is as large as possible. Thus, the mechanism may assume that the range of valid numeric box numbers is 0 to 9999, and the range of valid alphabetic box numbers is A to Z.

4. If the Record Type is "H" (High-rise), then the mechanism additionally tests whether the Apartment Number value is within the upper and lower bound values returned from the database for the current address. According to one feature, special processing is used to ensure a match. For example, If the Apartment Number is "2H", and the lower bound is "1A" and the upper bound is "3K," it is computationally difficult to determine whether "2H" is within the bounds. Therefore, in the preferred embodiment, the Apartment Number value and the bound values received from the database each are converted to base-36 numbers, and then compared. Base 36 is selected to account for digits "0" through "9" and characters "A" through "Z".

5. Whether a pre-directional term in the address information, if any, matches the pre-directional value in the current address record.

6. Whether a post-directional term in the address information, if any, matches the post-directional value in the current address record.

7. If the address line matches only one database record, processing is complete.

8. If there are multiple matches, and all are generally identical, then processing is complete. For example, if there are multiple entries for the same street in the database, which sometimes occurs because of data entry errors or data collection errors on the part of the database provider, then processing is complete.

If there are multiple different street suffixes (for example, "Hillcrest Road," "Hillcrest Drive," "Hillcrest Court," all associated with the same city and ZIP code), then the mechanism tests whether the parsed consumer address information includes a Suffix value that is among the multiple suffixes. If so, then a match occurs. If there is no match, then an error signal is generated. Thus, an exact suffix match is required in the case of such multiple records, because without requiring such a test, delivery errors could occur.

If there are multiple different pre-directional terms or post-directional terms, then the mechanism tests whether the parsed consumer address information includes a value that is among the multiple terms. If so, then a match occurs. If there is no match, then an error signal is generated. Thus, an exact match is required in the case of such multiple records, because without requiring such a test, delivery errors could occur.

9. If there are no matches of the address line to the database records, then the mechanism carries out additional tests, which may include the following. Each test comprises a modification of the address followed by a lookup in the database to match items in the database. If the lookup succeeds, then the mechanism concludes processing. If the lookup fails, then processing continues with the next test.

10. If there was a dash in atom number zero, as indicated by the Dash-In-Zero flag, then the address may be one that has a dash in the street number. That is, the element following the dash is not an apartment number, but is in fact part of a valid street number (e.g., having the form "100-400 Quetta Street"). In response, the first atom is converted to numeric form with a dash, the value of the Apartment Number field is appended, and the result is stored in the Street Number field. The corrected address is looked up in the database.

11. Buildings with suites are sometimes apartment buildings and sometimes not. Accordingly, if the converted address information has a Record Type of "H" (apartment building), then the Record Type value is changed to "S" (suite address), and the modified information is looked up.

12. If there was a single character near the beginning of the original address line, then the mechanism may have assigned that character to the Apartment Number field when it is actually a pre-directional value. For example, in the address " 123 N Main Street," the previous processing steps may have assigned the value "N" to the Apartment Number field. Thus, in response, the mechanism copies the Apartment Number value to the Pre-directional value, clears the Apartment Number field, and looks up the corrected information.

13. Test for different types of rural routes. In Canada, two different types of rural routes are used. Accordingly, in this step the mechanism converts the first type to the second type and re-checks the parsed information.

14. If no successful match in the database occurs after the foregoing tests, then the mechanism assumes that an error exists in the Street Name value. In response, the mechanism issues another database query that retrieves records for all streets associated with the ZIP Code value that was received from the consumer, and the matching process described above is re-executed against the records that are retrieved. Normally, this causes the mechanism to test the current values against a larger set of records. The mechanism then carries out one or more tests that modify the Street Name value and determine whether a match occurs with the modified value.

15. For example, if there is a pre-directional element and a post-directional element and the street name is a numeric value, then the address may be a grid-type address. In this context, a grid-type address is an address of the form "3000 West 1480 North," in which the address indicates a position in a grid of streets. In normal processing as described above, the element " 1480" would be converted in to the word expression " 1480th" which is erroneous and will not match the database. In response, the mechanism removes the suffix ("ST", "ND", "RD", "TH") from the street name and leaves a number as the street name. The modified address is then looked up in the database.

16. The mechanism may have introduced an error by converting a cardinal number word to a numeric value. For example, in the address "345 Hundred Oaks St.," converting the element "Hundred" to a numeric value "100" will cause an error. Therefore, the mechanism reverses such conversion, stores the cardinal number word in the Street Name field, and re-checks the address.

17. If the converted address contains the expression "State," and the original address contained the expression "St.", then it is possible that the expression "St." should have been converted to "Saint" rather than "State." Accordingly, the mechanism substitutes "Saint" for "State" in the address field and looks up the new information in the database.

18. The mechanism may have introduced an error by converting a street name suffix when, in fact, the true suffix has been omitted by the user. For example, in the address "204 Green Trail," the mechanism normally will convert the element "Trail" to "Trl." which is how the suffix "Trail" is expressed in the database. This may introduce an error when the true street name is "Green Trail Road" or the equivalent. In response, the mechanism will convert the suffix value to an expanded value and store the expanded value in the street name field. Then the data is re-checked against the database.

19. If there is still no match, then the mechanism returns an error value or an error message.

Other address checking steps may be carried out.

DETAILS OF AN EXAMPLE IMPLEMENTATION

A. COMMUNICATION PROTOCOL

In an embodiment, merchant server 108 communicates with commerce server 112 using Simple Commerce Messaging Protocol (SCMP), which is defined in the document http://search.ietf.org/internet-drafts/draft-arnold-scmp-01.txt. SCMP is an open standard that provides secure, authenticated commerce messaging capabilities between a sending agent's application to a receiving agent's server. The response by the receiving agent's server to the sending agent is a reply for the transaction represented by the set of data in the message. SCMP enables merchant server 108 and commerce server 112 to perform

on-line business transactions such that the merchant is fully authenticated, and the message cannot be repudiated.

In the preferred embodiment, SCMP messages requesting the use of commerce application 114 are sent from the merchant server 108 to commerce server 112. Each message contains fields that describe the application request and provide necessary information about the end user, the merchant, and the order. Preferably, SCMP messages are digitally signed and converted to ASCII format for transmission over a Hyper Text Transfer Protocol (HTTP) connection, enabling the messages to pass through firewalls and proxy servers.

Software elements that can send and receive SCMP messages are installed at merchant server 108 and commerce server 112. In operation, client 102 connects through network 106 to merchant server 108. A consumer that is operating client 102 and browser 104 uses the browser to display one or more electronic documents stored at merchant server 108 pages that display product and service information. When an order is entered, merchant server 108 exchanges one or more messages with commerce server 112 over network 106 to carry out order processing functions.

Each message provides commerce server 112 with information about the order processing function that is being requested. The information may include: verification of the merchant as a legitimate merchant for the requested commerce applications 114; identification of which commerce applications 114 to perform; all order and end user information required by the requested commerce applications; and detailed information about the products the consumer is ordering.

In one embodiment, the messages use the SCMP protocol, and the messages are created using scripts in C/C++ or Perl that call library functions to send the messages. In another embodiment, an interface to a commerce-ready third-party server ("commerce ready platform") is used, and the interface composes and sends the messages. Examples of commercially available commerce-ready third-party servers are Microsoft® Site Server 3.0, Commerce Edition; Microsoft® Active Server Pages; and BroadVision. One or more scripts are used to call the desired commerce applications 114 and interpret the results.

Within an SCMP message, separate fields specify information about (a) the commerce applications 114 that the merchant application 110 is requesting from

commerce server 112; (b) the order being placed, including line items of the order; and (c) the consumer who is placing the order. The commerce server 112 processes this information and returns other information as fields in an reply SCMP message.

A field in an SCMP message consists of the name of the field and a value. Thus, SCMP messages consist of a series of fields in name-value pairs, separated by newline characters. Two types of fields are recognized: order-level fields and offer-level fields.

When the merchant application 110 is working with function libraries of commerce server 112, then order level fields may be used within C/C++, Perl, or Java scripts. The merchant application 110 references the fields directly, specifying name-value pairs as described in later sections in this chapter. When the merchant application 110 is working with commerce-ready platforms, the merchant application uses an interface to specify field values, or where to find field values within a database or Web form. The merchant application 110 is presented with required and optional fields for each commerce application 114, and uses one or more scripts to call the applications 114 and interpret results.

A product description, known as a digital offer, is a single field that contains several sub-fields, referred to as offer-level fields. The offer-level fields describe a line item, or a product purchase, for an order. Digital offers allow a merchant to add or change its product mix with a minimum amount of re-programming of merchant application 110. Each product is represented to commerce server 112 as a stock-keeping unit (SKU) identifier. Merchant application 110 can send all information about a product with an end user's order without having to notify commerce server 112 in advance. For example, a merchant can do any of the following without notifying commerce server 112, by changing the digital offer: change the price for a product; hold a special promotion that will run for a specified period, after which the product returns to its standard price; change export restrictions on a product; change a risk threshold associated with a fraud screen application.

The digital offer also helps contain risk. Use of a digital offer ensures that all information in the digital offer, except for the quantity ordered, comes from database 111 and not from the consumer. In particular, use of a digital offer ensures that order parameters do not come from the HTML source code of the Web page to which the end user has access.

B. LOGICAL STRUCTURE AND INTERCONNECTION OF ADDRESS VERIFICATION MECHANISM AND COMMERCE SERVER

Address verification mechanism 116 may be implemented by creating an interface that sends SCMP messages requesting services from commerce applications 114.

In once embodiment, the interface may be developed by creating one or more scripts that send SCMP messages requesting one or more commerce applications 114. The scripts are written under Common Gateway Interface (CGI), Active Server Protocol (ASP), Internet Server Application Programming Interface (ISAPI), or Netscape Application Programming Interface (NSAPI) using pre-defined C/C++, Java, or Perl libraries. The libraries are commercially available from CyberSource Corporation.

Alternatively, the interface may be developed using object software such as CyberSource's Digital Commerce Component (DCC) that automates SCMP scripting. The DCC supports SCMP messaging and provides an interface to reduce development effort. An application developer may add appropriate scripts to interpret the results of a request for a customer.

In either alternative, the application developer provides particular information about the end user and order form that is relevant to the applications being requested. The commerce server 112 processes the requests and returns to the merchant server 108 a single message containing the results. Based on the information received, the merchant server 108 may accept or deny the end user's order and finish processing the transaction. This may include displaying results to the end user and storing order and returned data in database 111.

Merchant server 108 and merchant application 110 obtain a Merchant Key from commerce server 112, or its owner/operator, in advance of starting transaction processing operations. Before starting such transactions, it is assumed that the merchant associated with merchant server 108 has developed the business rules and all necessary data to support electronic commerce at its site, including but not limited to the following tasks: establishing a merchant account with a bank; identified and made agreements with any trading partners, such as distributors and resellers, and received approval and identifiers from commerce server 112 for such partners so they can be identified to commerce applications 114; created server programs or Web pages to accept product orders; created

server programs or Web pages to process orders; and set up customer and product information in database 111.

C. INVOCATION

In one embodiment, address verification mechanism 116 is used to verify delivery addresses for products ordered using online electronic commerce. In the preferred embodiment, address verification mechanism 116 is implemented as an application "ics_dav" that may be called from merchant application 110. The merchant application 110 may invoke the address verification mechanism 116 by including the application "ics_dav" in an SCMP request message. A request to carry out address verification is provided in an SCMP message from merchant application 110 to commerce server 112 and that includes all of the order fields and offer fields that are described below. The merchant application 110 then evaluates the returned fields to check for declined addresses or failures, as described further below. If the address is substantiated, the merchant may proceed to ship the product and may display appropriate information to the end user.

Table 1 describes the order-level fields that may be provided in a message from merchant application 110 to commerce server 112 that requests services by address verification mechanism 116.

TABLE 1 -- ORDER-LEVEL FIELDS

ORDER-LEVEL FIELD	DESCRIPTION	REQUIRED/ OPTIONAL
address1	Credit card billing street address as it appears in the credit card issuer's records. Up to 60 characters.	Required
city	Credit card billing city. Up to 50 characters.	Required
state	Credit card billing state. Must be the two-character postal-service abbreviation for the United States or Canada.	Required
zip	Credit card billing zip code.	Required
country	Credit card billing country. Must be a two-	Required

	character ISO 3166 country code.	
address2	Additional street address information.	Optional
ship-to addressl	Shipping street address. Up to 60 characters.	Optional
shipaddress2	Ship-to address2	Optional
shipcity	Ship-to city	Optional
shipstate	Ship-to state. Must be the two-character postal-service abbreviation for the United States or Canada.	Optional
shipzip	Ship-to ZIP.	Optional
shipcountry	Ship-to country. Must be a two-character ISO 3166 country code	Optional

Upon completion, address verification mechanism 116 returns values that indicate the results of its address verification operations. In the preferred embodiment, four values may be returned: an Order Number value; a Return Code value; a Ship Indication value; and an Error Message value.

The Order Number value is a tracking number generated by commerce server 112 that remains constant through the life of the order. The Return Code value indicates whether address verification succeeded. Preferably, the Return Code value is "1" if address verification is successful, "0" if the address has a problem, or "-1" if a system error occurs. The Ship Indication code indicates whether the order should be shipped to the address. When the Return Code value is 0 or -1, the Ship Indication value is "NSHIP". When the Return Code value is 1, the Ship Indication value is "YSHIP." The Error Message value is returned only when the Return Code value is equal to -1 or 0. The Error Message value contains a shorthand identification of the type of error that occurred.

D. ERROR PROCESSING

As described above, processing by the Address Verification mechanism may result in generating one or more error signals. In the preferred embodiment, each error signal is encoded according to a unique error value that is associated with a particular

type of error. Each error value is passed in a message from the Address Verification mechanism to the Commerce Server.

Table 2 is a list of error values and an error description that is associated with each error value.

TABLE 2 -- ERROR VALUES AND ERROR DESCRIPTIONS

ERROR VALUE	ERROR DESCRIPTION
NCOUNTRY	Country field or value unknown.
NCSZ	City or state, or zip unknown.
NCSZABBR	City or state, including abbreviated version, or ZIP code are unknown.
NDAVERR	Error performing address verification
NDB	Database and/or password is not recognized; unable to connect to database.
NPARSE	Street name unable to be parsed.
NSTREET	Street address unknown.

The Commerce Server passes a message that contains the error value to the merchant application 110 that requested the Commerce Server to activate the Address Verification mechanism. The merchant application 110 may display any appropriate response to the consumer. Normally, the error value is not displayed to the consumer, but interpreted by the merchant application 110, which presents an understandable response to the consumer.

ADVANTAGES; ADDITIONAL FEATURES OF ALTERNATE EMBODIMENTS

A. SOME ADVANTAGES

An address verification mechanism has been described which offers significant advantages over prior approaches. The mechanism is easily adapted to and readily parses non-United States addresses. The mechanism can be adapted to generate geo-coding as described above.

Further, the mechanism operates in real-time and therefore is ideal for the online commerce environment. The mechanism operates using standard open network protocols and does not require proprietary databases. The mechanism uses open interfaces and stateless processing which is ideal for use in the Internet environment. The merchant need not dedicate any storage space of its systems to support of the address verification mechanism. The mechanism may execute asynchronously and in parallel with other online commerce operations.

B. OTHER FEATURES AND FUNCTIONS

1. BAR CODE GENERATION

The mechanism may be implemented using many other features and functions. For example, based on the parsed address information, the mechanism could generate and return digital information that represents an address bar code in compliance with U.S. Postal Service bar code standards. A calling program, or the merchant, could use the bar code information to print compliant bar codes on mailing labels of packages prior to shipment. The bar code information could be returned in an additional field of the response message.

2. HIGH RISK ADDRESS FILTERING

The mechanism can be enhanced with high-risk address filtering, whereby the merchant is alerted that an address is "high risk" using a dedicated response code or error code. An example of a high risk address is an address known to be associated with a private, for-profit mail box provider, a General Delivery address, etc.

3. SHIPPING TYPE FILTERING

The mechanism can carry out shipping type filtering. For example, assume that the consumer has requested the merchant to ship by United Parcel Service (UPS). UPS does not ship to Post Office Box addresses. The merchant could provide a Shipper Type value to the address verification mechanism when it is called. When the address verification mechanism determines that the address information provided by the consumer specifies a Post Office Box, the mechanism could test the value of the Shipper Type field, and if it is "UPS" or the equivalent, the mechanism would generate an error message. This test could be conducted before any database lookup is carried out. Such a

lookup would be unnecessary because the merchant would want to reject the shipment even if the Post Office Box is valid.

HARDWARE OVERVIEW

FIG. 4 is a block diagram that illustrates a computer system 400 upon which an embodiment of the invention may be implemented. Computer system 400 includes a bus 402 or other communication mechanism for communicating information, and a processor 404 coupled with bus 402 for processing information. Computer system 400 also includes a main memory 406, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 402 for storing information and instructions to be executed by processor 404. Main memory 406 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 404. Computer system 400 further includes a read only memory (ROM) 408 or other static storage device coupled to bus 402 for storing static information and instructions for processor 404. A storage device 410, such as a magnetic disk or optical disk, is provided and coupled to bus 402 for storing information and instructions.

Computer system 400 may be coupled via bus 402 to a display 412, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 414, including alphanumeric and other keys, is coupled to bus 402 for communicating information and command selections to processor 404. Another type of user input device is cursor control 416, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 404 and for controlling cursor movement on display 412. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

The invention is related to the use of computer system 400 for automatically verifying address information. According to one embodiment of the invention, automatically verifying address information is provided by computer system 400 in response to processor 404 executing one or more sequences of one or more instructions contained in main memory 406. Such instructions may be read into main memory 406 from another computer-readable medium, such as storage device 410. Execution of the

sequences of instructions contained in main memory 406 causes processor 404 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 404 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 410. Volatile media includes dynamic memory, such as main memory 406. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 402. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor 404 for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 400 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus 402. Bus 402 carries the data to main memory 406, from which processor 404 retrieves and executes the instructions. The instructions received by main memory 406 may optionally be stored on storage device 410 either before or after execution by processor 404.

Computer system 400 also includes a communication interface 418 coupled to bus 402. Communication interface 418 provides a two-way data communication coupling to a network link 420 that is connected to a local network 422. For example, communication interface 418 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 418 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 418 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 420 typically provides data communication through one or more networks to other data devices. For example, network link 420 may provide a connection through local network 422 to a host computer 424 or to data equipment operated by an Internet Service Provider (ISP) 426. ISP 426 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 428. Local network 422 and Internet 428 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 420 and through communication interface 418, which carry the digital data to and from computer system 400, are exemplary forms of carrier waves transporting the information.

Computer system 400 can send messages and receive data, including program code, through the network(s), network link 420 and communication interface 418. In the Internet example, a server 430 might transmit a requested code for an application program through Internet 428, ISP 426, local network 422 and communication interface 418. In accordance with the invention, one such downloaded application provides for automatically verifying address information as described herein.

The received code may be executed by processor 404 as it is received, and/or stored in storage device 410, or other non-volatile storage for later execution. In this manner, computer system 400 may obtain application code in the form of a carrier wave.

SCOPE OF DISCLOSURE

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A method for verifying address information, the method comprising the computer-implemented steps of:
receiving address information associated with an electronic commerce transaction;
parsing the address information to produce a hypothesized address;
using the hypothesized address to automatically determine in real-time whether the address information is valid; and
creating and storing a validity value representing whether the address information is valid.

2. The method as recited in Claim 1, further comprising verifying in real-time a city value, a state value and zip code value from the address information against a commercial database of standard address information.

3. The method as recited in Claim 1, wherein parsing the address information to produce a hypothesized address further comprises:
removing extraneous spaces and extraneous punctuation from the address information;
determining whether the address information is a general delivery type; and
breaking the address information into atoms and applying one or more validation tests to the atoms when the address information is not a general delivery type.

4. The method as recited in Claim 3, wherein breaking the address information into atoms further comprises one or more of the following steps:
removing any text following the symbol "/" when "/" is not part of a one half indication;
replacing any atom with a space when the atom does not contain either an alphanumeric character or a pound sign;

breaking a first atom at the hyphen when the first atom contains a hyphen;
breaking any atom at commas when the atom contains commas.

5. The method as recited in Claim 3, wherein applying one or more validation tests to the atoms further comprises creating and storing the hypothesized address based on result values from one or more tests selected from among:
converting number words to numerals;
processing rural route addresses;
processing street numbers;
processing apartment indicators;
processing pre-directional elements;
processing "Floor" designations;
splitting an atom into two atoms at a pound sign;
storing the values in all the atoms in a Street Name value when there is no street number atom;
performing tests for post office box addresses;
processing post-directional elements;
processing suffix indicators;
converting ordinals into numerals;
performing tests to determine foreign language street type designations.

6. The method as recited in Claim 3, wherein applying one or more validation tests further comprises identifying and storing a Street Name value by performing one or more of the following steps:
storing in the Street Name value the values of any atoms that have not been identified as other values;
identifying and storing state route and highway designations appearing in the atoms;
identifying one or more pre-directional elements from the Street Name value and storing each pre-directional element in a pre-directional value;
creating and storing a numeric suffix when the Street Name value consists of only a number.

7. The method as recited in Claim 6, wherein applying one or more validation tests further comprises hypothesizing and storing an Apartment Number value when one or more atoms are left unprocessed, a penultimate atom contains either a suffix indicator, a pre-directional element , or a post-directional element, and a last atom is not associated with any other element.

8. The method as recited in Claim 5, further comprising determining a record type based on a Country value in the address information.

9. The method as recited in Claim 5, further comprising identifying a Street Name value by performing one or more of the following steps:

creating and storing in the Street Name value a directional word associated with a pre-directional value;

creating and storing in the Street Name value "PO Box" when the Post Office Box value is not null;

creating and storing in the Street Name value alphabetic characters that are identified as representing a Street Number;

storing a Suffix value in the Street Name value when the address information contains only two atoms.

10. The method as recited in Claim 2, further comprising:

obtaining from a database all city names and state names associated with the zip code value of the address information;

determining whether the state value of the address information matches any of the state names from the database; and

returning an error message when the state value of the address information does not match any of the state names from the database.

11. The method as recited in Claim 10, further comprising:
determining whether the city value of the address information matches any of the city names from the database when the state value of the address information matches any of the state names from the database; and
terminating verification of address information when the city value of the address information does not match any of the city names from the database.

12. The method as recited in Claim 10, further comprising selecting the database based on a Country value in the address information.

13. The method as recited in Claim 10, further comprising;
retrieving and storing the city name from the database based on the zip code value of the address information when the city value of the address information does not match any of the city names from the database; and
further parsing the address information to produce a hypothesized address using the received city name.

14. The method as recited in Claim 10, further comprising returning a city error message in real-time when the city value from the address information does not match any of the city names from the database.

15. The method as recited in Claim 10, further comprising:
requesting corrective action in real-time when the city value of the address information does not match any of the city names from the database;
receiving a corrected city name value; and
parsing the address information to produce a hypothesized address using the corrected city name value.

16. The method as recited in Claim 15, wherein requesting corrective action in real-time further includes returning the city name from the database based on the zip code value of the address information as a suggested city name value.

17. The method as recited in Claim 10, further comprising:
removing all non-alphabetic characters from the city value of the address information to result in a modified city value;
determining whether the modified city value matches the city name from the database while allowing a predefined tolerance for typographical errors;
determining whether the modified city value matches a city name abbreviation in a predefined list of city name abbreviations when no match is found between the modified city value and the city name from the database; and
storing a full city name corresponding to the city name abbreviation for the modified city value when a match is found between the modified city value and the city name abbreviation in the predefined list of city name abbreviations.

18. The method as recited in Claim 17, further comprising:
determining whether the modified city value matches a city name abbreviation constructed from the city name from the database when no match is found between the modified city value and the city name from the database; and
storing a full city name corresponding to the city name abbreviation constructed from the city name from the database for the modified city value when a match is found between the modified city value and the city name abbreviation constructed from the city name from the database.

19. The method as recited in Claim 1, wherein using the hypothesized address to automatically determine in real-time whether the address information is valid further comprises:
retrieving, from a database, address records that have values matching a Street Name value and a zip code value of the hypothesized address;
performing a first set of tests to determine whether any of the address records from the database match the hypothesized address;
terminating verification of address information and returning a success signal when there is only one address record from the database that matches the hypothesized address; and

terminating verification of address information and returning a success signal when there are multiple address records from the database that match the hypothesized address and the multiple address records are identical.

20. The method as recited in Claim 19, further comprising:

performing a second set of tests when there are multiple address records from the database that match the hypothesized address and the multiple address records are not identical;

performing a third set of tests when there are no address records from the database that match the hypothesized address; and

retrieving from the database all address records that have the same value as the zip code value of the hypothesized address before performing a fourth set of tests when the third set of tests fails to find the address record that matches the hypothesized address; and

returning an error message in real-time when the fourth set of tests fails to find the address record that matches the hypothesized address.

21. A method for processing address information, the method comprising the computer-implemented steps of:

receiving address information associated with an electronic commerce transaction;

parsing the address information to generate hypothesized address information; and

using the hypothesized address information to automatically generate digital information representing map coordinate values that identify a geographic location of a receiving party associated with the address information.

22. The method as recited in Claim 21, further comprising applying fraud control to the address information by:

verifying in real-time a city value, a state value, and a zip code value in the address information to produce a verified city name, a verified state name and a verified zip code; and

creating and storing a risk flag value when the verified city name, verified state name, and verified zip code is determined to represent a high-risk address.

23. The method as recited in Claim 22, further comprising:

generating a longitude value and a latitude value as the map coordinate values based on the verified city name, verified state name and verified zip code; and

providing the map coordinate values to a shipper of goods for the electronic commerce transaction.

24. A computer-readable medium carrying one or more sequences of one or more instructions for verifying address information, the one or more sequences of one or more instructions including instructions which, when executed by one or more processors, cause the one or more processors to perform the steps of:

receiving address information associated with an electronic commerce transaction;

parsing the address information to produce a hypothesized address;

using the hypothesized address to automatically determine in real-time whether the address information is valid; and

creating and storing a validity value representing whether the address information is valid.

25. A system for verifying address information, the system comprising:

a memory;

one or more processors coupled to the memory; and

at least one processor configured to carry out the steps of:

receiving address information associated with an electronic commerce transaction;

parsing the address information to produce a hypothesized address;

using the hypothesized address to automatically determine in real-time whether the address information is valid; and

creating and storing a validity value representing whether the address information is valid.

26. A method for processing address information, the method comprising the computer-implemented steps of:

a memory;

one or more processors coupled to the memory; and

at least one processor configured to carry out the steps of:

receiving address information associated with an electronic commerce transaction;

parsing the address information to generate hypothesized address information; and

using the hypothesized address information to automatically generate digital information representing map coordinate values that identify a geographic location of a receiving party associated with the address information.

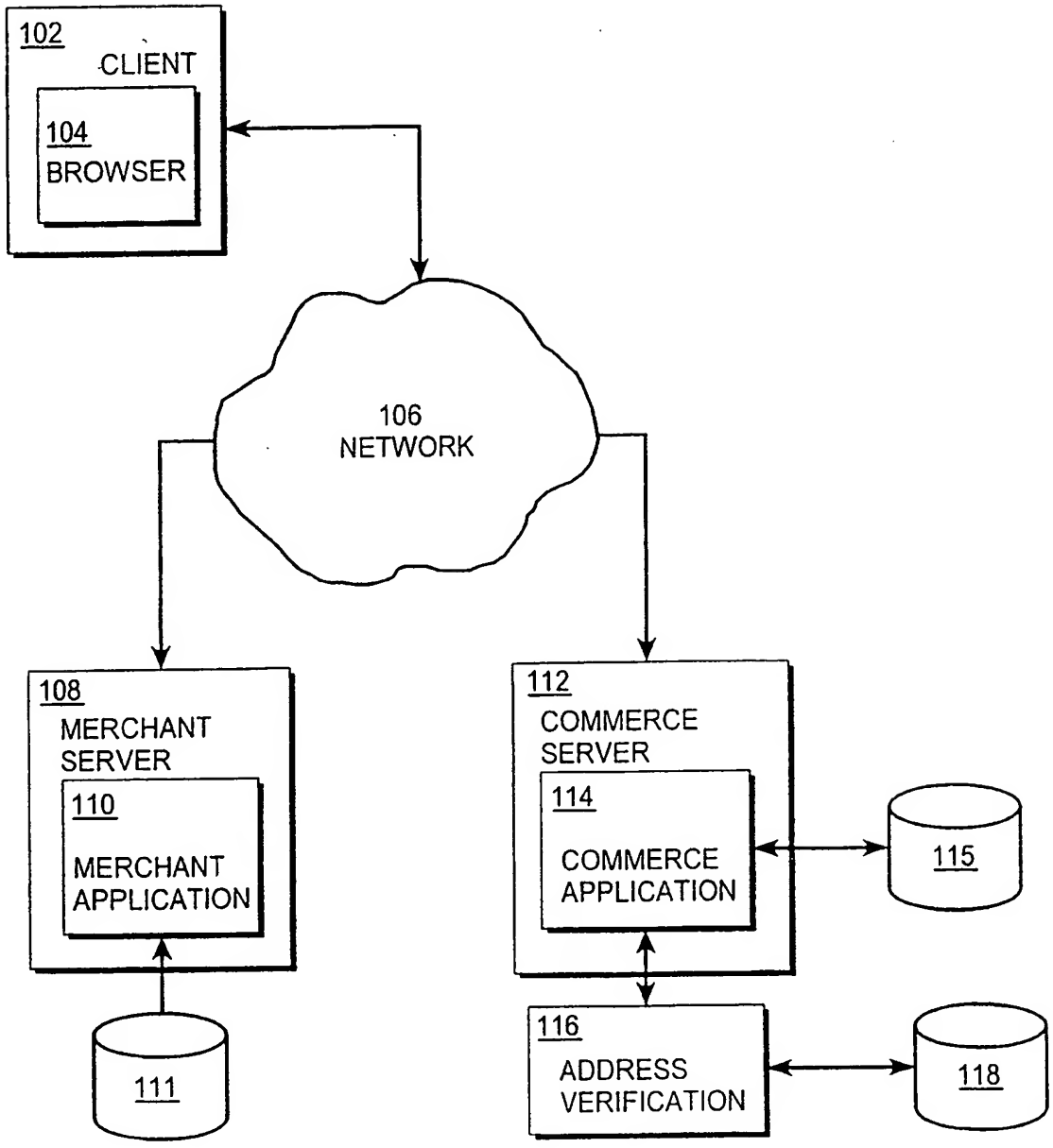
102
CLIENT
104
BROWSER
106
NETWORK
108
MERCHANT
SERVER
110
MERCHANT
APPLICATION
111
112
COMMERCE
SERVER
114
COMMERCE
APPLICATION
115
116
ADDRESS
VERIFICATION
118

Fig. 1

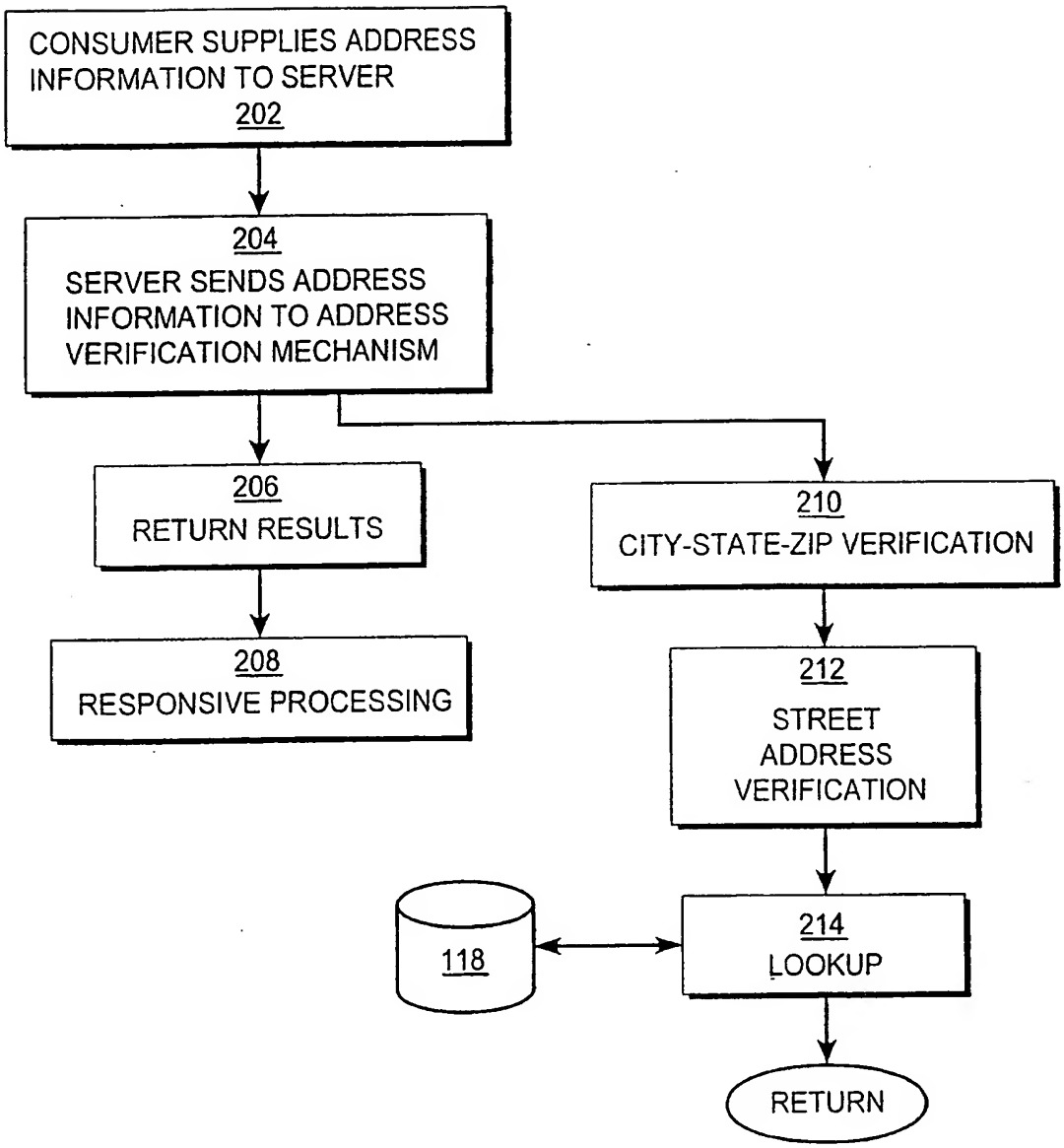
CONSUMER SUPPLIES ADDRESS INFORMATION TO SERVER
202
204
SERVER SENDS ADDRESS INFORMATION TO ADDRESS VERIFICATION MECHANISM
206
RETURN RESULTS
208
RESPONSIVE PROCESSING
210
CITY-STATE-ZIP VERIFICATION
212
STREET ADDRESS VERIFICATION
118
214
LOOKUP
RETURN

Fig. 2

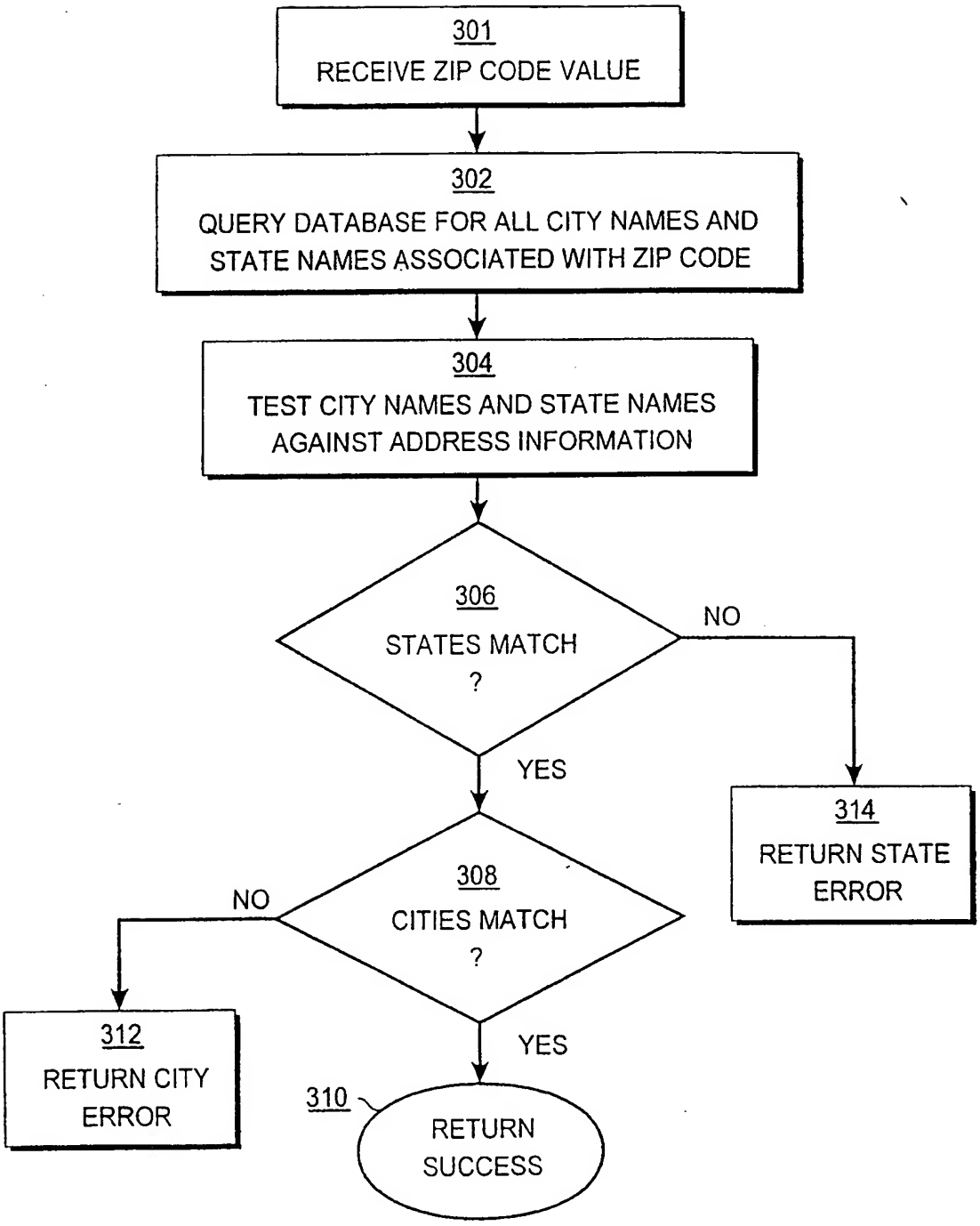
301
RECEIVE ZIP CODE VALUE
302
QUERY DATABASE FOR ALL CITY NAMES AND STATE NAMES ASSOCIATED WITH ZIP CODE
304
TEST CITY NAMES AND STATE NAMES AGAINST ADDRESS INFORMATION
306
STATES MATCH ?
NO
YES
314
RETURN STATE ERROR
308
CITIES MATCH ?
NO
YES
312
RETURN CITY ERROR
310
RETURN SUCCESS

Fig. 3

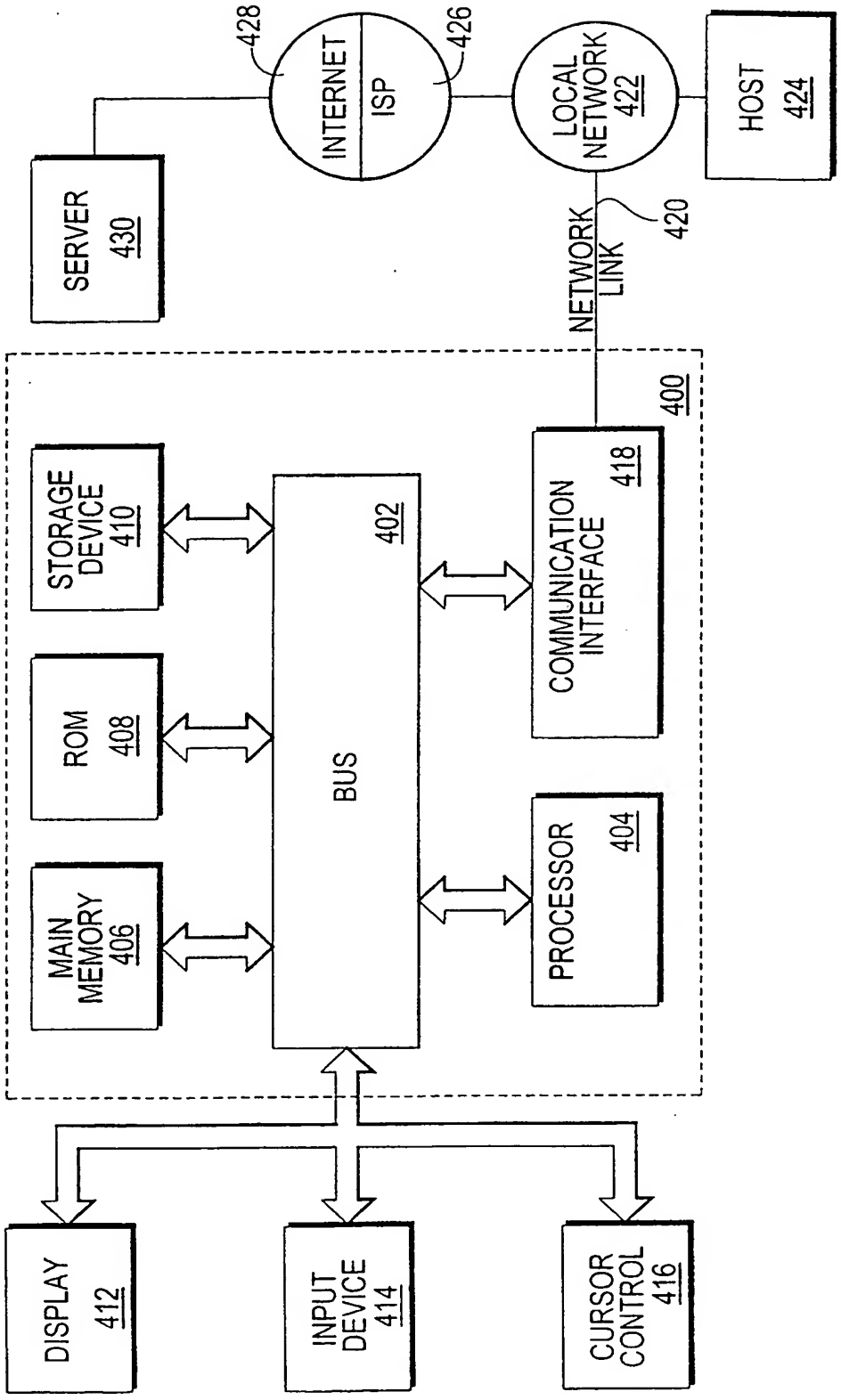
428
INTERNET
ISP
426
LOCAL NETWORK 422
HOST 424
SERVER 430
NETWORK LINK
420
400
STORAGE DEVICE 410
ROM 408
MAIN MEMORY 406
BUS
402
COMMUNICATION INTERFACE 418
PROCESSOR 404
DISPLAY 412
INPUT DEVICE 414
CURSOR CONTROL 416

Fig. 4

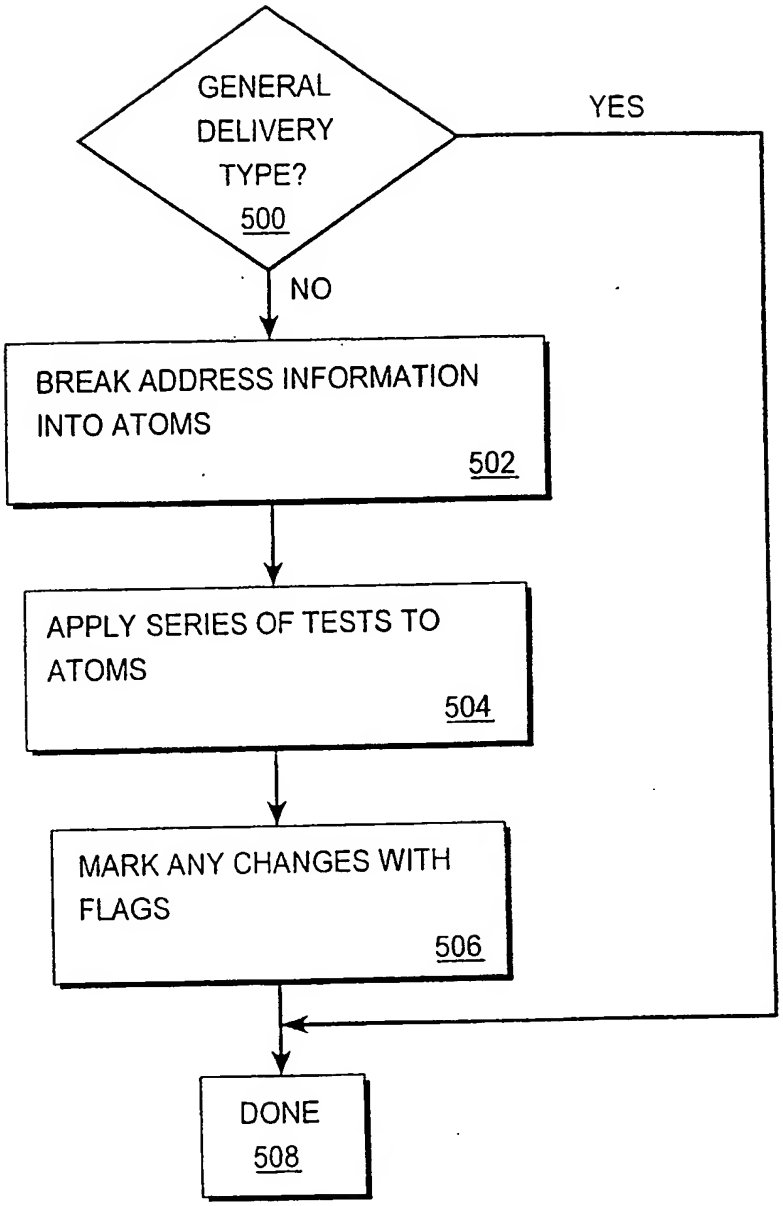
GENERAL DELIVERY TYPE?
500
YES
NO
BREAK ADDRESS INFORMATION INTO ATOMS
502
APPLY SERIES OF TESTS TO ATOMS
504
MARK ANY CHANGES WITH FLAGS
506
DONE
508

Fig. 5

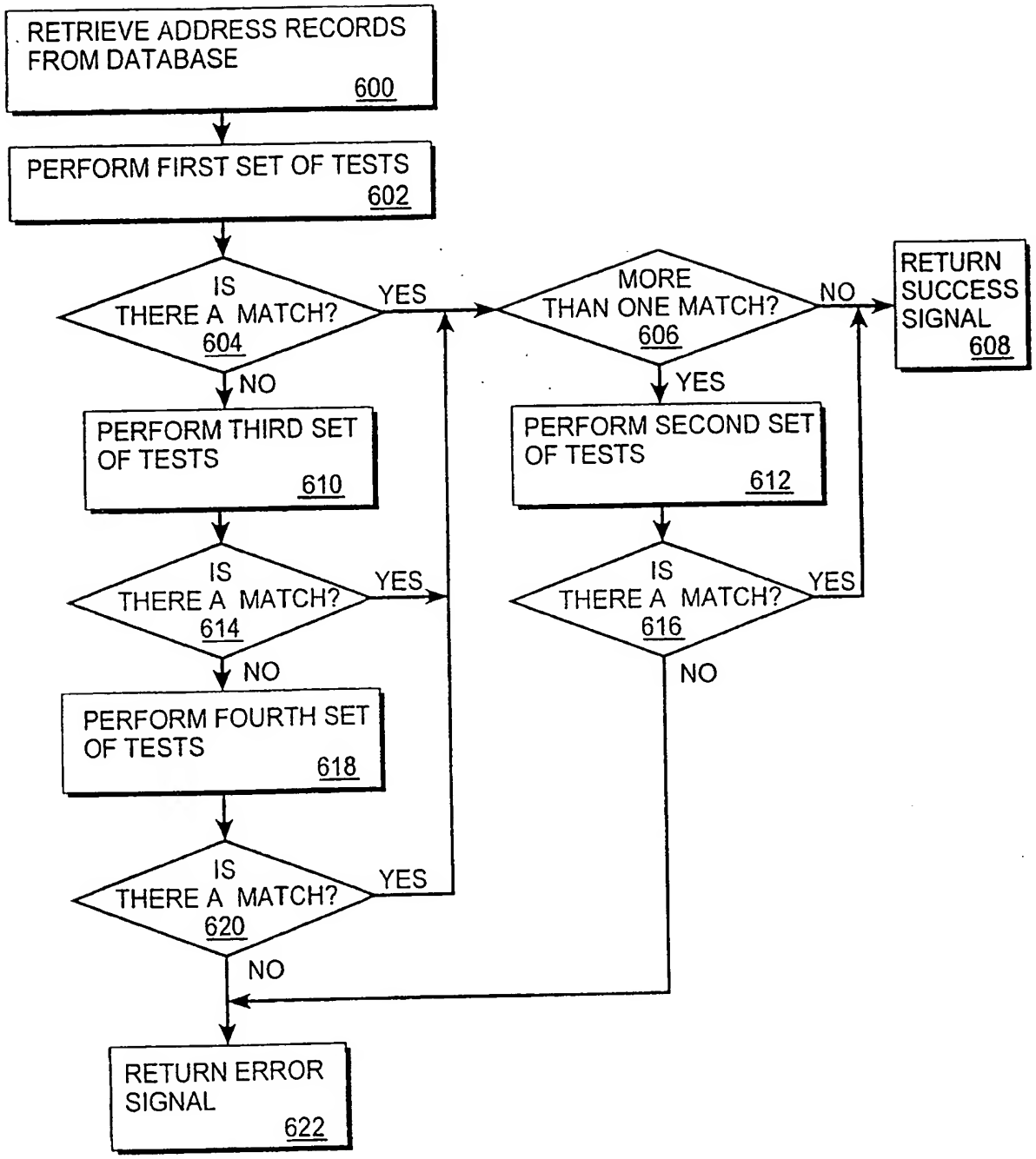
RETRIEVE ADDRESS RECORDS FROM DATABASE
600
PERFORM FIRST SET OF TESTS
602
IS THERE A MATCH?
604
YES
NO
MORE THAN ONE MATCH?
606
NO
YES
RETURN SUCCESS SIGNAL
608
PERFORM THIRD SET OF TESTS
610
PERFORM SECOND SET OF TESTS
612
IS THERE A MATCH?
614
YES
NO
IS THERE A MATCH?
616
YES
NO
PERFORM FOURTH SET OF TESTS
618
IS THERE A MATCH?
620
YES
NO
RETURN ERROR SIGNAL
622

Fig. 6

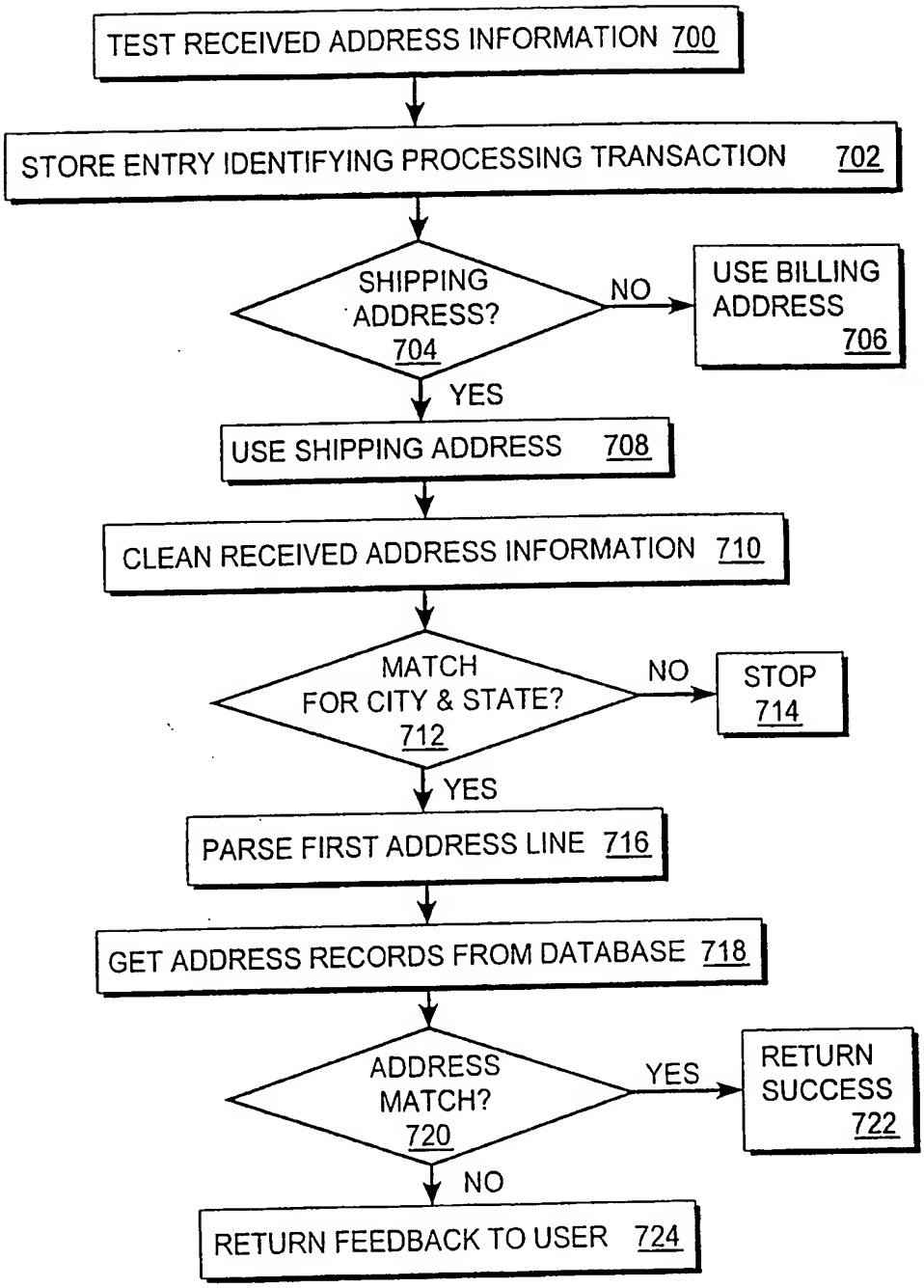
TEST RECEIVED ADDRESS INFORMATION 700
STORE ENTRY IDENTIFYING PROCESSING TRANSACTION 702
SHIPPING ADDRESS? 704
NO
USE BILLING ADDRESS 706
YES
USE SHIPPING ADDRESS 708
CLEAN RECEIVED ADDRESS INFORMATION 710
MATCH FOR CITY & STATE? 712
NO
STOP 714
YES
PARSE FIRST ADDRESS LINE 716
GET ADDRESS RECORDS FROM DATABASE 718
ADDRESS MATCH? 720
YES
RETURN SUCCESS 722
NO
RETURN FEEDBACK TO USER 724

Fig. 7

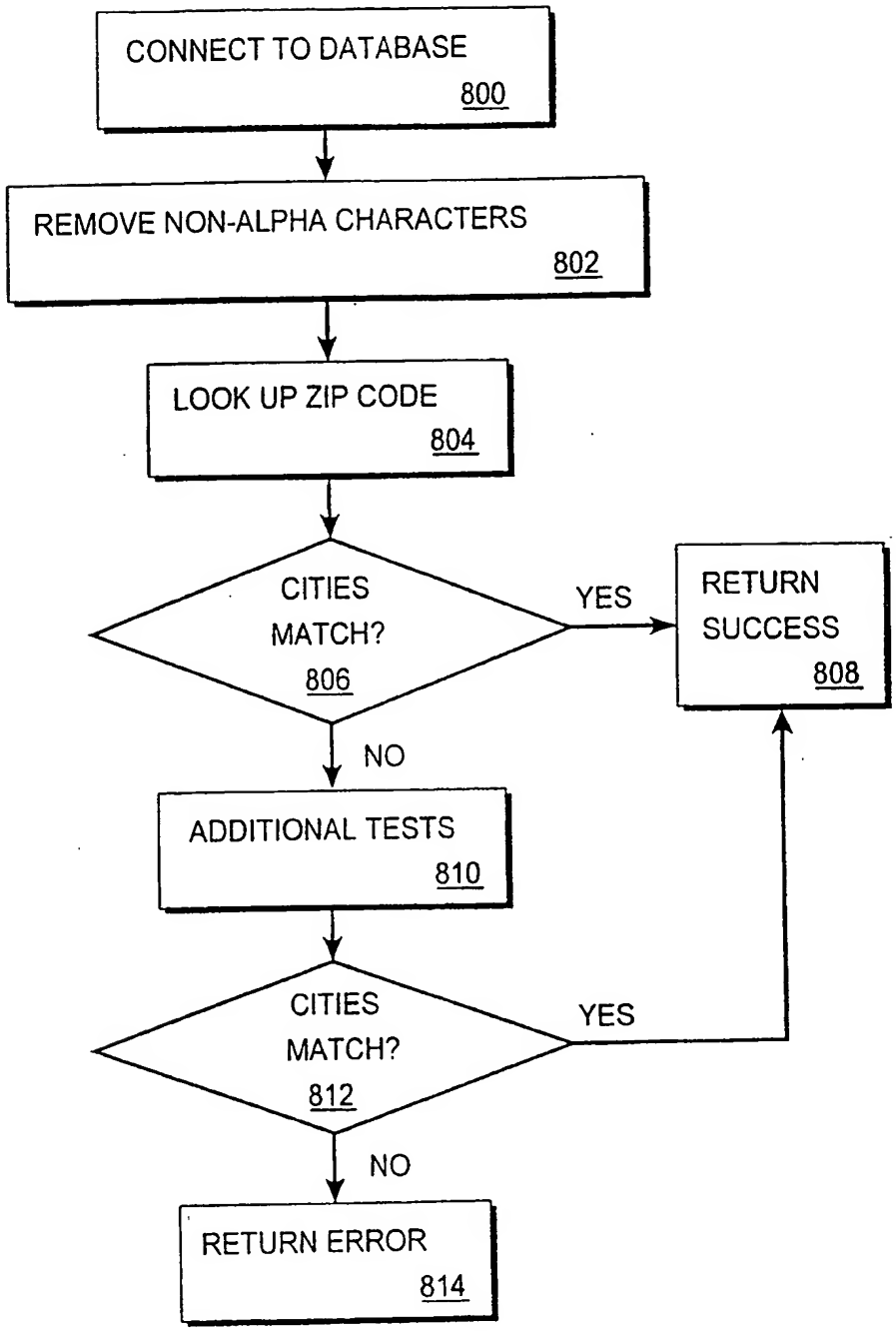
CONNECT TO DATABASE
800
REMOVE NON-ALPHA CHARACTERS
802
LOOK UP ZIP CODE
804
CITIES MATCH?
806
YES
RETURN SUCCESS
808
NO
ADDITIONAL TESTS
810
CITIES MATCH?
812
YES
NO
RETURN ERROR
814

Fig. 8